CoLink: A Programming Framework for Decentralized Data Science
Literature Review

Xiaoyuan Liu
xiaoyuanliu@berkeley.edu

Tianchen Liu
tianchenliu@berkeley.edu

Bryce Wong
brywong@berkeley.edu

1 Introduction

Data collaboration is a common need in many research fields including finance, health care, cybersecurity, and many others. To protect data privacy in such collaborations, many existing protocols and systems support certain types of computations without sharing the original data input across different participants. However, there are many deployment challenges for using existing solutions in the real world and there lacks a universal framework with a consistent data collaboration workflow.

In this work, we propose to investigate an efficient programming abstraction to support decentralized data science. It provides a unified interface for the user, storage, communication, and computation. Extending gRPC, we propose to simplify the development of multi-party protocols and allow implementations in different programming languages to work together consistently. With a unified interface that increases potential data contributors, our framework has the potential to enable larger-scale decentralized data collaboration and unlock the true value of data.

1.1 Deployment Difficulty

Many existing works focus on the design of the security protocols and evaluate their performance via simulations. Building and deploying systems for real-world problems based on these protocols requires extra effort. We identify three major deployment difficulties as follows.

- **Practicability.** Compared with protocol simulations that focus on correctness and performance evaluation, a real-world privacy-preserving system needs great engineering efforts to be practical. In most solutions, different participants need to be able to discover and identify each other. They often need to build secure and authenticated connections for communication, store and transmit data, and perform various computations. Building such a system requires the management of authenticated users and their roles, database, network communication, computation runtime, and many other resources.
- **Compatibility.** Often time, the implementation of one protocol would require the functionality of many other protocols. For example, federated learning often requires secure aggregation as its building block to merge the gradients in a privacy-preserving way. Secure aggregation might further require synchronization for randomness which can be supported by various protocols. However, different protocols are often implemented with different library dependencies in different programming languages, making it hard for integration. Even protocols implemented in the same programming language might be using different data formats and communication libraries. Poor compatibility not only introduces additional integration effort but also makes the replacement of protocol with similar functionality difficult. A solution using a migrated protocol cannot directly benefit from an ungraded building block without another round of migration.
- **Publicity.** Sometimes the designer of a new system might want protocol with certain functionality and security property as a building block. However, knowing and understanding the terms to describe the protocol requires comprehensive background knowledge in the field of cryptography. Even with the precise term to describe the requirement, a survey for a proper implementation also takes time. On one hand, not all research papers include the protocol implementation code. On the other hand, sometimes there are multiple implementations available with various differences that can be only discovered after testing. There lacks a unified way to describe the functionalities and security guarantees of various protocols and an index that supports searching and comparison for them.

1.2 Proposed Solution

To accelerate the development of security protocols and overcome the deployment obstacles, we propose to build CoLink, a programming framework for real-world privacy-preserving solutions. CoLink targets the following functionalities.

- To improve practicability, CoLink should provide a minimal set of abstractions that is useful in most protocols. It should include a public-key-based user system with corresponding authentication and access control mechanisms. It should integrate an in-memory key-value database with potential persistence. It should handle communications securely. Based on the implementation of these subsystems, it should provide a concise but flexible programming interface to express various types of multi-party data collaborations that can be used in different programming environments.
- To improve compatibility, CoLink should enable the complete execution of a protocol to be conducted through multiple languages and environments. It should provide mechanisms to encapsulate existing protocols with well-defined behavior and security properties and manage the virtualization for different programming runtimes. It should allow developers to reuse existing protocols as building blocks easily while designing new protocols.
- To improve publicity, CoLink should provide an indexing mechanism to list out existing protocols supported in its interface and describe the participants, functionalities, and security properties of those protocols in a structural way. It should allow developers to easily find protocols that match their requirements and set up the corresponding programming runtime without much effort. In addition to managing the protocols, it should also provide a registry mechanism for user and data discovery so that researchers can also find collaborators with data that matches their interests.

To achieve the above functionalities, next we propose a system design and select some candidate technologies. CoLink includes three parts: a core service that runs on the cloud and acts on behalf of the user, programming language-specific SDKs for developers with different purposes, and frontend for system management.

- **Core service.** The core service manages both the computations and relevant data. The system users upload the data to the core service before requesting computations with other participants. The core service communicates with other core services, manages the computation requests, and allows users to decide whether to accept, ignore, or reject the requests. Once all participants' core services confirm the computation plan, they prepare the data and pass the control to specific protocol implementation. We propose to use mutual TLS-guarded gRPC [10] for the interface and in-memory key-value database like Redis [24] for storage. We also propose to add a message queue component like RabbitMQ [30] for subscription to simplify full-duplex transmissions.
- **SDK.** We identify two types of developers that might benefit from a system like CoLink. Privacy-preserving application developers take an off-the-shelf existing protocol and mainly focus on scenario-specific development. In comparison, protocol designers might use multiple existing protocols as building blocks and implement a new protocol that can be used by any CoLink users. We propose to provide separate SDKs to two groups. For application developers, we provide an application SDK (SDK-A) that allows them to interact with core services to manage core storage, schedule new computation instances, manage computation requests, and get results. For protocol developers, we provide a protocol SDK (SDK-P) which is a super-set of SDK-A and also covers the functionalities of creating new protocols like integration with the core to process certain protocol runs.
- **Frontend.** In addition to the SDK for interacting with the CoLink system, we also propose to build a frontend that uses CoLink SDK-A in JavaScript to display system status information and simplify certain manual operations like confirming a computation request. To improve publicity, we also propose to build an index for sample protocols supported by CoLink, so that application developers can find protocols that match their needs.

We expect to make the following contribution.

1. We propose a programming framework to simplify the deployment of protocols for decentralized data science. We will discuss a list of abstractions that we identify as the most common and important for building real-world solutions.
2. We will develop of prototype to demonstrate the usefulness of our system design. Specifically, we will implement a core service that manages users, storage, communication, and computations. We will also provide SDKs in different languages to show how to easily combine two protocols implementations written in different languages.
3. We will also include several sample decentralized data science applications built upon CoLink.

2 Related Work

Although there is no previous work targeting the exact same direction, there are many works focusing on relevant or orthogonal goals that might provide insights for the design of CoLink. In this literature review, we first discuss a list of decentralized data science applications that are potentially interesting to support. Then, we discuss existing work on decentralized abstractions which is relevant to the construction in CoLink.

2.1 Decentralized Data Science Apps

Federated Learning. Federated learning [15, 37, 11] is one of the most well-known decentralized data applications. It trains a machine learning model using data from different participants without exchanging the original data instances. With secure aggregation protocols [6] for merging gradients, it achieves better privacy guarantees by hiding privacy-leaking updates from the individual. There are also existing works [5, 26, 25, 13] focusing on system construction to improve communication efficiency or improving the overall learning algorithms [14]. In addition to learning logistic regression or neural-network-based models, tree-based models [7, 36, 28] are also often considered, especially when input features are separate vertically across participants.

Federated learning solution developers can potentially benefit from CoLink. Because the user and communication abstractions in CoLink are helpful in orchestrating complicated communication structures between coordinators in FL servers and the clients who hold the data. With compatibility with existing secure aggregation protocol implementation, the protocol designer can focus on the machine learning part and easily add and replace secure aggregation building blocks. The final packaged CoLink-compliant protocol could be used by the developer who does not understand protocol details but still want to use it in their software products.

Federated Analytic. As mentioned in the early federated learning paper [5], federated analytics [33] that aggregate various statistics can also be an important decentralized data application with value for real-world deployment. For example, Google uses federated analytics to evaluate its word prediction model on the client's GBoard app [1]. Instead of focusing on the learning of a machine learning model, it focuses on "collaborative data science without data collection" [1] which presents an overlap with the goal in this work.

In our work to support "decentralized data science", we want to focus more on the asymmetric role assignment in the decentralized setting and also support protocols with various network topology in malicious settings.

Secret Sharing and Secret Management. Secret sharing [27] is a common building block for many multi-party computation-based protocols. In real-world systems, it can be also used for storage [22] when combined with other techniques. To further protect the confidentiality of the shared secrets, there is also research focusing on the refresh of the secrets [8] for long-term protection.

Considering its wide usefulness, we plan to integrate the secret sharing scheme as a sample protocol in CoLink. Furthermore, we also plan to support the additive operations for the shared secrets to demonstrate how the computation and the storage abstractions work in CoLink.

Private Query Processing. In addition to storing the data, there are also existing works targeting to allow query functionalities to data that is securely stored or secret-shared. CryptDB [21] allows one party to store encrypted data on another party but still be able to perform queries on those data. Splinter [34] focuses more on query obliviousness and performs queries on secret shared data on non-colluding servers to hide query access patterns from the data owners. There are also many other solutions [20, 31, 3] with various settings but all target to support a privacy-preserving database for one or multiple data owners.

Because we expect CoLink users to store their data for collaboration at their core services, it would be beneficial to also allow users to make queries on data from others. We plan to integrate protocols that support private query processing to support this functionality.

Privacy-preserving Machine Learning. In addition to the model training in federated learning, there are also existing works focusing on privacy-preserving machine learning [18] that present a more strict limitation for information being exchanged during both the inference and the training of the model. As the performance of the privacy-preserving inference becoming practical [4, 23, 17, 32, 29], we also plan to investigate the possibility of providing model and instance abstractions to support it in CoLink.

2.2 Decentralized Abstractions

Many existing works also propose to help the deployment of decentralized data science and provide various abstractions. Here we discuss a few relevant solutions that are either similar to the construction in CoLink or could be used together with CoLink for additional functionalities.

Global data plane [19] also provides storage and communication-related abstraction in a distributed set-

ting but focuses more on the data provenance tracking. CALYPSO [12] uses distributed ledgers as its storage building block and focuses on monetizing auditable data access with access control. Memri [16] targets to boost private and trustworthy use of data and allows users to manage their data themselves. Gretel [9] improve privacy for accessing data using differential privacy and data synthesis and provide users with API to simplify development.

There are also works focusing on strengthening privacy and security in different decentralized applications. Data Capsule [35] allows automated privacy analysis for regulation compliance using a policy language.Viaduct [2] builds a compiler for secure distributed programs and maps an application program to the language in its MPC or ZKP backends.

References

[1] Federated analytics: Collaborative data science without data collection. `https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html`. Accessed: 2020-05-27.

[2] ACAY, C., RECTO, R., GANCHER, J., MYERS, A. C., AND SHI, E. Viaduct: an extensible, optimizing compiler for secure distributed programs. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (2021), pp. 740–755.

[3] BATER, J., ELLIOTT, G., EGGEN, C., GOEL, S., KHO, A. N., AND ROGERS, J. Smcql: Secure query processing for private data networks. *Proc. VLDB Endow. 10*, 6 (2017), 673–684.

[4] BOEMER, F., COSTACHE, A., CAMMAROTA, R., AND WIERZYNSKI, C. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography* (2019), pp. 45–56.

[5] BONAWITZ, K., EICHNER, H., GRIESKAMP, W., HUBA, D., INGERMAN, A., IVANOV, V., KIDDON, C., KONEČNÝ, J., MAZZOCCHI, S., MCMAHAN, B., ET AL. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems 1* (2019), 374–388.

[6] BONAWITZ, K., IVANOV, V., KREUTER, B., MARCEDONE, A., MCMAHAN, H. B., PATEL, S., RAMAGE, D., SEGAL, A., AND SETH, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 1175–1191.

[7] CHENG, K., FAN, T., JIN, Y., LIU, Y., CHEN, T., PAPADOPOULOS, D., AND YANG, Q. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems 36*, 6 (2021), 87–98.

[8] DESMEDT, Y., AND JAJODIA, S. Redistributing secret shares to new access structures and its applications. Tech. rep., Citeseer, 1997.

[9] GRETEL AUTHORS. gretel.ai.

[10] GRPC AUTHORS. grpc.

[11] KAIROUZ, P., MCMAHAN, H. B., AVENT, B., BELLET, A., BENNIS, M., BHAGOJI, A. N., BONAWITZ, K., CHARLES, Z., CORMODE, G., CUMMINGS, R., ET AL. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning 14*, 1–2 (2021), 1–210.

[12] KOKORIS-KOGIAS, E., ALP, E. C., GASSER, L., JOVANOVIC, P., SYTA, E., AND FORD, B. Calypso: Private data management for decentralized ledgers. *Cryptology ePrint Archive* (2018).

[13] KONEČNÝ, J., MCMAHAN, H. B., YU, F. X., RICHTÁRIK, P., SURESH, A. T., AND BACON, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[14] LI, T., SAHU, A. K., ZAHEER, M., SANJABI, M., TALWALKAR, A., AND SMITH, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems 2* (2020), 429–450.

[15] MCMAHAN, B., MOORE, E., RAMAGE, D., HAMPSON, S., AND Y ARCAS, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (2017), PMLR, pp. 1273–1282.

[16] MEMRI AUTHORS. memri.io.

[17] MISHRA, P., LEHMKUHL, R., SRINIVASAN, A., ZHENG, W., AND POPA, R. A. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium (USENIX Security 20)* (2020), pp. 2505–2522.

[18] MOHASSEL, P., AND ZHANG, Y. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)* (2017), IEEE, pp. 19–38.

[19] MOR, N. *Global data plane: A widely distributed storage and communication infrastructure*. University of California, Berkeley, 2019.

[20] PAPPAS, V., KRELL, F., VO, B., KOLESNIKOV, V., MALKIN, T., CHOI, S. G., GEORGE, W., KEROMYTIS, A., AND BELLOVIN, S. Blind seer: A scalable private dbms. In *2014 IEEE Symposium on Security and Privacy* (2014), IEEE, pp. 359–374.

[21] POPA, R. A., REDFIELD, C. M., ZELDOVICH, N., AND BALAKRISHNAN, H. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (2011), pp. 85–100.

[22] RAMAN, R. K., AND VARSHNEY, L. R. Distributed storage meets secret sharing on the blockchain. In *2018 Information Theory and Applications Workshop (ITA)* (2018), IEEE, pp. 1–6.

[23] RATHEE, D., RATHEE, M., KUMAR, N., CHANDRAN, N., GUPTA, D., RASTOGI, A., AND SHARMA, R. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020), pp. 325–342.

[24] REDIS LTD. Redis.

[25] ROMANINI, D., HALL, A. J., PAPADOPOULOS, P., TITCOMBE, T., ISMAIL, A., CEBERE, T., SANDMANN, R., ROEHM, R., AND HOEH, M. A. Pyvertical: A vertical federated learning framework for multi-headed splitnn. *arXiv preprint arXiv:2104.00489* (2021).

[26] RYFFEL, T., TRASK, A., DAHL, M., WAGNER, B., MANCUSO, J., RUECKERT, D., AND PASSERAT-PALMBACH, J. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017* (2018).

[27] SHAMIR, A. How to share a secret. *Communications of the ACM 22*, 11 (1979), 612–613.

[28] TIAN, Z., ZHANG, R., HOU, X., LIU, J., AND REN, K. Federboost: Private federated learning for gbdt. *arXiv preprint arXiv:2011.02796* (2020).

[29] TRAMER, F., AND BONEH, D. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287* (2018).

[30] VMWARE, INC. Rabbitmq.

[31] VOLGUSHEV, N., SCHWARZKOPF, M., GETCHELL, B., VARIA, M., LAPETS, A., AND BESTAVROS, A. Conclave: secure multi-party computation on big data. In *Proceedings of the Fourteenth EuroSys Conference 2019* (2019), pp. 1–18.

[32] WAGH, S., TOPLE, S., BENHAMOUDA, F., KUSHILEVITZ, E., MITTAL, P., AND RABIN, T. F: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies 2021*, 1 (2021), 188–208.

[33] WANG, D., SHI, S., ZHU, Y., AND HAN, Z. Federated analytics: Opportunities and challenges. *IEEE Network* (2021).

[34] WANG, F., YUN, C., GOLDWASSER, S., VAIKUNTANATHAN, V., AND ZAHARIA, M. Splinter: Practical private queries on public data. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)* (2017), pp. 299–313.

[35] WANG, L., NEAR, J. P., SOMANI, N., GAO, P., LOW, A., DAO, D., AND SONG, D. Data capsule: A new paradigm for automatic compliance with data privacy regulations. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 3–23.

[36] WU, Y., CAI, S., XIAO, X., CHEN, G., AND OOI, B. C. Privacy preserving vertical federated learning for tree-based models. *arXiv preprint arXiv:2008.06170* (2020).

[37] YANG, Q., LIU, Y., CHEN, T., AND TONG, Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST) 10*, 2 (2019), 1–19.